Agility and Respect

Neville Holmes
University of Tasmania

The Agile values are ethical values.

My programming experience started when programs were hardware built from wires plugged into control panels. For three decades after that I was involved with computing projects of many kinds.

So far, throughout this millennium, I have occasionally browsed articles and reports on Agile programming with a strong feeling of déjà vu. Recently, the 2009 Australian Software Engineering Conference invited me to give a keynote talk. Their formal focus was to be "Agile, the New Mainstream" (aswec2009.itee.uq.edu.au). Recalling my déjà vu, I decided my talk would be on "The Prehistory and the Future of Agility" (eprints.edu.au).

THE TALK

To prepare the talk, I first consulted Wikipedia and then the Agile Manifesto (agilemanifesto.org), which gives four values. These form the basis of agility and are expressed as preferences. I chose to treat each preference in the light of my early experience, though not in the Manifesto's sequence.

For ***Working software*** *over comprehensive documentation*, I talked of my early programming experience. For ***Customer collaboration*** *over contract negotiation*, I drew on my experience with data processing management. For ***Individuals and interactions*** *over processes and tools*, my later experience with research projects formed the basis. For ***Responding to change*** *over following a plan*, my normal role of working directly with users proved relevant. The discussion of each value ended with a challenge to the computing profession, all challenges I had already touched on in essays in this column.

The conference was nostalgic and stimulating. Perhaps it was the interaction with attendees that caused me later to see that these challenges were in fact ethical as much as they were professional. Therefore, the following sections repeat the challenges but pick out the ethical aspect of each Agile value.

RESPECTING TALENT

Personality is compounded of talents. Because they are compounded differently, despite the efforts of consumer culture to impose uniformity, people differ. This applies to both their private and public lives. It is a basic ethical value to respect, even to promote, those differences.

Back when scientists called computing automatic data processing, the data handlers who operated the various machines developed their logical talents through their work. The more experienced handlers programmed the machinery and documented the procedures.

However, the logic needed to program the control panels of the more complex calculators and to write the abstract programs that complemented the panels of the late-1950s and early-1960s computers led to specialization.

Very early in the process, managers discovered that some people had an instinctive talent for programming, so such people became the specialists—well before any computing degrees or diplomas were offered. As stored-program computers became more affordable and thus more popular, the demand for programmers became greater. People with a distinct talent for programming are not common, as anyone who has tried to teach programming to unselected students knows, so programming aptitude tests were successfully used to find good candidates (*The Profession*, Nov. 2004, pp. 120, 118-119).

In the mid-1960s, as computers became more capable, less talented programmers were often found to be more useful as systems analysts who would work in partnership with programmers and take responsibility for documentation.

Now software engineering aims to be a branch of engineering, but is finding it difficult to be accepted

Continued on page 98

 Published by the IEEE Computer Society

Continued from page 100

as such. The problem is that other branches sensibly use the skills and talents of technicians to ensure the success of their professional work. Software engineering doesn't; it won't let go of programming.

The challenge then is for software engineers to respect the talent of programmers and to use them as technicians, and so respect their own distinct talents by distracting them less and developing them more (*The Profession*, Sept. 2002, pp. 104, 102-103). Engineers look after people's interests; technicians look after the engineers' interests by using tools and machinery. These technicians deserve respect: Their talents are just as important as those of engineers.

The issue of documentation is peripheral. While different aspects of an undertaking need different documentation by different agents, the communication between engineer and technician should be close and continual, as befits a technical partnership.

RESPECTING SERVICE

The nature of our society mandates that some people work for other people. In ethical societies this is a relationship of master and servant or employer and employee. The servant does work for the master, and this work is best done in a relationship of mutual respect. In particular, the servant respects the master's interest in the outcome of the work he does.

Fifty years ago, the popular management structure for commercial entities was called staff/line/service. The line people did the work, the staff people managed the work and the workers, and the service people provided support for the staff and line. The data processing department was a service department that looked after the entity's data and printed out data as, when, and how needed. This organization and attitude held up into the early days of stored-program computers.

Problems arose when computers became more capable in both storing data and processing it. These problems were basically political. Companies spent more money both on computing equipment and on people to operate, program, and exploit the equipment.

Problems arose when computers became more capable in both storing data and processing it.

DP management began to separate the programmers and systems analysts organizationally, however. The systems analysts specified in great detail the program suites to be developed before these were handed over to the programming section to implement. This, in turn, disrespectfully dissociated the programmers from interacting with people—theirs was a backroom job tucked away from real life. This also meant that only large projects were feasible, which led to projects failing either because of faults in the specifications (and changes to specifications made work on such projects horrendous) or because requirements had changed during the project's implementation.

Because of their increasing budgets, DP management began to rise in the entity's hierarchy and wasn't content to be seen as providing a service. They held themselves to be builders of systems; staff management, whose ranks they had joined, believed them. Later, as the projects became increasingly larger, along with the failures, consultants and contractors were brought in, which made the situation still more complex. The contracts involved were, in my opinion, more intended to avoid blame than to achieve success.

This problem arose because of a change in attitude. The project became the focus instead of the service the project was supposed to provide. The investment was seen as being in the DP department's programs and computers, rather than in maintaining the entity's data and the provision of services based on that data.

For a commercial or government entity to use digital technology effectively, it must turn its focus from the DP/IT department's software and hardware to the entity's data (*The Profession*, June 2006, pp. 100, 98-99). Its computing people must be able to collaborate closely and continually to provide needed services promptly and cheaply. For the computing profession to support this properly, it must promote the creation of data engineers alongside and perhaps even instead of software engineers.

RESPECTING EXPERTISE

An expert is someone with special skill and knowledge in a particular area. Any genuine expert deserves respect, and learned professions have codes of ethics intended to ensure that the respect is preserved for the profession as a body. Special skill and knowledge in the professions is gained through education and postgraduate experience.

When there were relatively few computers, skill and knowledge in computing was special and deserved respect, but the professional development has since been mixed and questionable. Early expertise at universities lay in science and engineering, and it was in those schools that computer-science education first developed. Commercial use of stored-program computers came later and was simpler because, in contrast to technical applications, commercial applications applied simple programs to large files of data. Somewhat belatedly, professional education in computing sought a place for itself in

business schools by offering degrees in "information systems."

Now that computers are ubiquitous, computing skills are no longer special. In the typical university, every department has its computing laboratories. In developed countries, schools have computers galore, and most homes have them as well.

Computing professionals are losing respect and disappearing behind geeks, nerds, and hackers. University education in computing as a specialty is struggling to find students and content.

To regain respect, the computing profession and its educators must realize that computing professionals need to combine their special skill and knowledge with another professional area's special skill and knowledge. The computing profession cannot exist for computing's sake alone. It must work in close partnership with other professions, and computing professionals must be educated to work in close partnership with other professionals (*The Profession*, Jan. 2007, pp. 116, 114-115).

RESPECTING USERS

In the staff/line/service organizational model, it's the line workers who produce the goods or serve the customers. Thus it's the line workers who bring in the revenue.

With the growing strength and power of the DP department, the line worker or end user missed out. The systems analysts or consultants decided what services were delivered to the line worker, with perhaps some consultation with staff managers. Unfortunately, management often doesn't understand what end users do and need. I clearly remember in my days as a systems engineer having to act as a kind of ambassador to upper management for line workers.

The effect has been that, notionally to avoid failure, management has striven to use modern interactive digital technology to drive line workers as automata. This shows no respect for line workers and, moreover, robs them of self-respect. Tragedy can result.

At breakfast on the day I gave my keynote talk at ASWEC, I read in the morning paper of a lad lost in the bush who died because the emergency operators he repeatedly called on his mobile phone refused to act when he couldn't give them a street address. "The operators had been 'fixated' on asking for a street address because it was in accordance with their training and the steps they were to follow within the computer program." (tinyurl.com/nru3no)

The challenge is to enable end users rather than disable them—to show them respect (*The Profession*, May 2008, pp. 92, 90-91).

This could be accomplished by providing end users with the training to program their computer in simple ways. In the early 1960s, some companies had their systems programmers start doing this by maintaining a library of macrodefinitions that let users write procedures in simple and obvious code. Unfortunately, DP management's political growth, coupled with the adoption of Cobol, scotched this.

Interestingly, IBM successfully marketed small business computers in the 1970s by providing a template programming tool called RPG (Report Program Generator) that let such businesses exploit their computers without needing a DP department.

The four challenges I've described are dramatic indeed: To establish a culture of programming technicians; to turn administrative computing from a major project culture to a data maintenance and service culture; to couple professional computing education with other professional education; and to empower line and other employees to take responsibility for their use of computers rather than to be driven by them.

Significantly, the questions asked after my keynote talk largely supported my challenges. The main opposition focused on the first challenge and seemed to come from senior academics. In my opinion, though, this is the most straightforward and overdue of the four reforms, and its adoption would simplify achieving the other three. Indeed, if either software or data engineering are to become accepted branches of engineering, professional bodies like the IEEE and national computer societies must concertedly press for the recognition of programming as a trade and seek the cooperation of other professional engineering bodies.

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing and Information Systems. Contact him at neville.holmes@utas.edu.au.*